

**1. Title: A method and system for linking user input with streaming data**

The invention is a method and system to automatically link user input with a playback or recording of streaming data through time-correlated, event-generated data pointers. The invention can be used to link meeting minutes with an audio recording of the meeting. A closed captioning system could also use this invention to get a rough alignment between the transcript and the audio or video recording [Pat5].

**2. Abstract**

Our invention is a method and system to automatically link user input with a pre-recorded playback or live recording of streaming data via time-stamped event pointers. The system automatically detects pre-defined trigger events, pairs an event label with a pointer to the data that caused the event, then time stamps and records the data in a master file. User input is thus linked to the streaming data by the nearest-in-time trigger event recorded for the streaming data.

**3. Inventorship**

Davis Pan  
US Citizen  
48 Hodge Road, Arlington, MA 02474, USA  
Badge: 170764  
Phone: (617) 551-7652  
Fax: (617) 551-7650  
Cost Center: YAQ  
Email: pan@crl.dec.com  
Manager: Rishiyur Nikhil

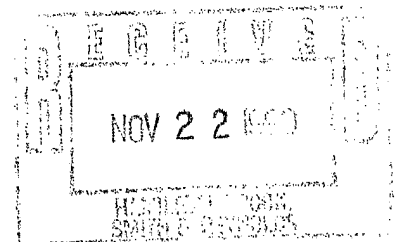
Jim Rehg  
US Citizen  
106 Quincy Street, Arlington, MA 02476, USA  
Badge: 326517  
Phone: (617) 551-7645  
Fax: (617) 551-7650  
Cost Center: YAQ  
Email: rehg@crl.dec.com  
Manager: Rishiyur Nikhil

**3.1. DATE OF CONCEPTION**

The initial idea for this system originated from a conversation between the co-inventors on February 18, 1999. The co-inventors developed most of the details of the invention on the next day, February 19.

**3.2. DATE OF REDUCTION TO PRACTICE**

As of September 1999, no experimental tests have been performed. Reduction of this invention to practice has not yet been planned.



### 4.1. PURPOSE

The invention is a method and system to automatically link user input with a playback or recording of streaming data through time-correlated, event-generated data pointers. There are many practical applications of such a system:

- The invention could link meeting minutes with an audio recording of the meeting.
- A closed captioning system could use this invention to get a rough alignment between the transcript and the audio or video recording [Pat5].
- The invention could be incorporated in a video annotation system.

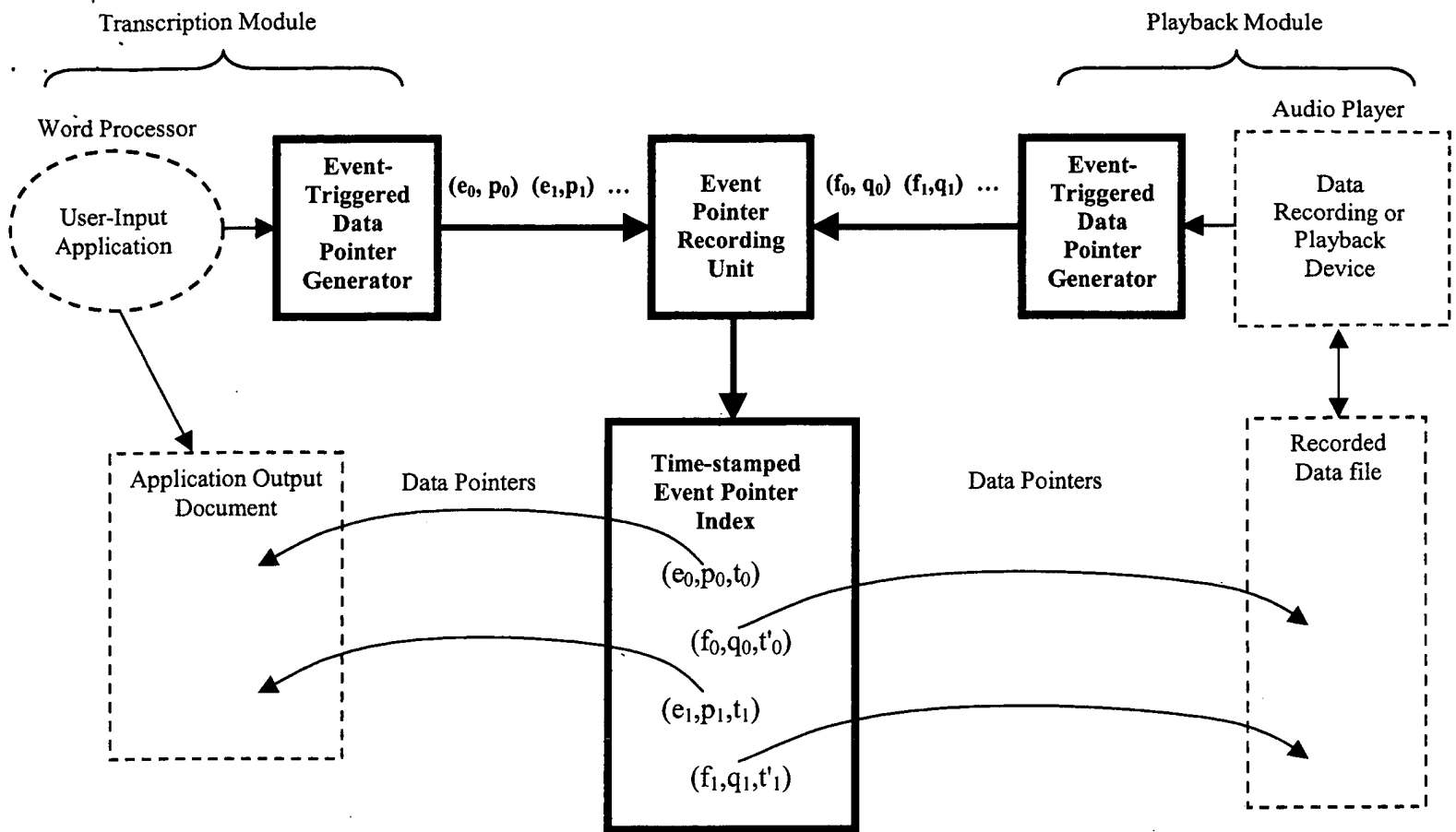
### 4.2. BACKGROUND MATTER

Finding a desired section within a recording of streaming data (e.g. an audio or video tape recording) is typically difficult even for the person who made the recording. Without some additional information, it is impossible for someone who has never witnessed the recorded data to find a desired section short of playing back the data from the start. Often it is desirable to correlate the contents of an audio or video recording with additional user data, such as notes or annotations. For example, an audio or video recording of a presentation could be correlated with a set of notes taken during the course of the talk.

Systems for event logging provide a basic correlation capability. The UNIX utility `syslogd` provides a facility for time-stamped logging of user-defined events. Events generated by multiple sources are logged into a central file along with their time-stamps. Events can be correlated by comparing time-stamp values. Similarly, in an X windows system, user events can be logged along with their time stamps as a record of the user's activities.

There is a variety of prior art describing event logging systems and their applications. The patent, *Arrangement for recording and indexing information* [Pat1], addresses the specific problem of correlating hand-written notes to a reel-type audio recording. The invention is a device that automatically correlates the position of the hand-written notes on a page with the position of the audio tape on the reel. The patent, *Interactive system for producing, storing and retrieving information correlated with a recording of an event* [Pat2], is a more general approach to linking information with a recording. However this invention forces the user to explicitly request a 'time zone' event in order to correlate subsequent user input to a given section of recorded data. The patent, *Method and apparatus for recording and analyzing an interaction log* [Pat3], describes a system with an event detecting means that triggers a state detecting means wherein the state detecting means detects at least 2 parameters, position and time, of objects within a display window. These parameters are recorded and linked with the triggering event, but not with another synchronous data stream. The patent, *Computer graphics data recording and playback system with a VCR-based graphic user interface* [Pat4], is limited to recording of an X-windows session wherein X-windows commands, states, and events are linked via time-stamps. Similarly, the patent *Computerized court reporting system* [Pat6] is limited to synchronizing stenographic annotations with a video recording of a trial.

Event logging methods assume that a data-pair consisting of an event label and a time-stamp is sufficient to determine the relationship between time sequences of actions. This is often the case for computer system events or X window events. However, this approach is problematic for streaming media data. An event logging approach is possible if one assumes that streaming data is produced or recorded at a known rate. Given the data rate and a particular time stamp it is possible to determine a position within the data stream.



**Figure 1:** System architecture. Primary components of invention are drawn with solid bold lines. The use of the invention in an audio transcription system is illustrated with dotted lines.

However, in many important applications the streaming rate can vary asynchronously, as when a media source such as a tape player is paused, reversed, or advanced frame-by-frame using a thumbwheel.

An alternative to correlating time-stamped events with a continuous data stream is to break a data stream up into chunks, each associated with a particular event. An example of this approach is the "record narration" feature of Microsoft Powerpoint97. Each slide in a presentation can be linked to a separate audio file which will be played whenever that slide is rendered.

In cases where a multimedia data stream contains an audio channel, speech technology can be used to correlate the data stream with a text transcript. By performing speech recognition on the audio channel it may be possible to correlate words in a transcript with positions in the data stream. However this approach is limited to data streams with speech audio on which speech analysis is successful.

### 4.3. THE HOWs

The invention is a system for automatically correlating one or more unsynchronized streams of data by means of predefined, time-stamped events with associated data pointers. Figure 1 shows a block diagram of the invention. The system consists of a plurality of Event-Triggered Data Pointer Generators and an Event Pointer Recording Unit. Each generator unit is designed to detect a set of user-defined events and communicate the occurrence of each event along with an associated data pointer. Each generator produces a sequence of pairs of the form  $(E_i, P_i)$  where  $E_i$  is an event label,  $P_i$  is a data pointer and  $i$  denotes the position in the sequence.

The dotted lines in Figure 1 illustrate the use of the invention for on-line audio transcription. In this

application, a user listens to an audio signal and transcribes it into text in real-time. The user has a means to slow-down or speed-up the rate of audio playback as needed to aid in transcription. This is described in a companion patent filing [Pat7]. Note that the playback device could be an external analog tape deck which is physically distinct from the computer system on which the transcription is being entered. There are two generator units. They take their inputs from the word-processing software on which the transcript is entered and from the external tape deck (through an appropriate hardware interface).

The trigger events for transcription, labeled  $e_i$  in the figure, are generated whenever the user types a new word in the word-processing application. There are three types of transcription trigger events, called *append*, *insert*, and *delete*. Append events are generated whenever a word is added to the end of the current transcript. Insert and delete events are generated whenever the user backs up inside previously typed text and inserts or deletes a word. The associated data pointers, labeled  $p_i$  in the figure, give the absolute position of each word within the document. In our preferred embodiment this is a word identifier which is incremented in units of 100 to accommodate insertions and deletions. A typical sequence of event-pointer pairs might be: (append, 300), (append, 400), (insert, 350), (append 500), etc. The transcript generator unit can easily produce word identifiers for append events by incrementing the most recent identifier by 100. Identifiers for insert events can be produced by computing the average of the identifiers for the two words which border the insertion.

The playback generator takes its input from the analog tape deck. Playback trigger events, labeled  $f_i$  in the figure, are produced whenever there is a change in the playback rate of the audio data. In a simple instantiation there would be five trigger events corresponding to the functions *play*, *stop*, *fast-forward*, *rewind*, and *pause*. The data pointers are labeled  $q_i$  in the figure. They identify the absolute position in the audio source, called the audio position, at which a trigger event occurred. In a simple instantiation a pointer could be a burned-in time-code on the tape which is read by a high-end deck such as a Betacam. A typical sequence of event-pointer pairs might be: (play,  $T_0$ ), (rewind,  $T_1$ ), (play,  $T_2$ ), (fast-forward,  $T_3$ ), etc. where the  $T_i$ 's are time-codes.

The Event Pointer Recording Unit assigns a universal time-stamp to each event-pointer pair, resulting in a triple of the form  $(E_i, P_i, t_i)$  where  $t_i$  denotes the time-stamp for the  $i$ th triple. The triples are recorded in the Time-stamped Event Pointer Index, where they can be ordered on the basis of their time-stamp value. In the example shown in the figure, two sequences of triples of the form  $(e_0, p_0, t_0)$ ,  $(e_1, p_1, t_1)$ , ... and  $(f_0, q_0, t'_0)$ ,  $(f_1, q_1, t'_1)$ , ... are logged in the index. In this example we assume that each time-stamp  $t'_i$  is matched to time-stamp  $t_i$ . If each stream of event pairs is recorded separately, the matching process can be accomplished with a simple merge sort. The time-stamp matching process serves to correlate the unsynchronized data streams represented by the sequences of pointers  $p_i$  and  $q_i$ . In the transcription application, words in the transcript are matched to positions on the audio tape.

An important point is that in many applications such as transcription, preprocessing of the index is necessary before the time-stamp matching process can be performed. There are two categories of preprocessing: *event filtering* and *data pointer interpolation*. Event filtering is necessary whenever insertions or deletions are being made into a data stream. This occurs in the transcription module above, since words are inserted and deleted into the transcript in real-time as the user corrects mistakes or omissions. Deletion events are filtered by matching their data pointer to the data pointer of a previous append event, and then removing both events from the index. Likewise, the data pointers associated with insertion events are assigned a time-stamp which is consistent with their position in the stream. This is done by changing the time-stamp for insertion events to the average of the time-stamp values for the data pointers that border the insertion. As an example consider the following sequence of triples produced by the Event Pointer Recording Unit from the output of the transcript generator and stored in the index: (append, 500, 20), (append 600, 25), (append, 700, 35), (delete, 500, 40), (insert, 650, 50). After filtering, this sequence becomes: (append, 600, 25), (insert, 650, 30), (append, 700, 35).

The second preprocessing operation is data pointer interpolation. This is performed in order to generate a regular stream of data pointers from a sparse set of trigger events. This is necessary in the playback module

of the transcription system, since the goal is to correlate the transcript with the audio stream at a much finer temporal scale than that of the tape deck trigger events. Each event such as play or fast-forward changes the playback speed of the tape. Thus this speed is known between events. Given the time-stamps for two events, the elapsed time that separates them can be divided into regular intervals and assigned additional time-stamps. Given the data pointers at the event occurrence and the playback speed, it is simple to interpolate the pointer values at the boundary across the time intervals. The result is a dense set of data pointers at regular intervals. The size of the sampling intervals are chosen depending on the desired resolution in correlating the text to the audio.

The inclusion of data pointers in the index is a key part of the invention that distinguishes it from prior art such as event logging systems. Data pointers are necessary in dealing with a data stream, such as words in a transcript, that is not contiguous with real-time. The presence of insertions and deletions removes any simple correlation between stream position and real-time. Correlation can be restored through post-hoc analysis using the stored data pointers.

Data pointers are also useful in providing robustness to system latency. This can be illustrated using the example of audio transcription. Suppose the task is to correlate user input with an audio stream. The Event-Triggered Data Pointer Generator for the audio playback device produces a sequence of pairs consisting of a playback event label (play, fast forward, rewind, stop, pause) and its associated data pointer (e.g. the audio position). These data provide a complete record to the point in the audio signal where the event occurred. Latency at the Event Pointer Recording Unit will only effect the accuracy of the temporal correlation between the user input events and the audio playback events. It will not effect the positional accuracy of the data pointer. In contrast, an event logging system would simply time-stamp the event labels. In these systems, audio positions must be deduced from the time-stamp and the audio playback rate associated with each event. Thus any difference between the recorded time-stamp and the actual time at which an event occurred will produce errors in the audio position. These errors will be further compounded by latencies in logging user events.

The ordered index that results from time-stamp matching serves to cross-link multiple data streams, permitting several different kinds of access. For example, given a desired time, the closest time-stamp for each data stream can be identified. Using the associated data pointers, the corresponding data stream positions associated with that time can be identified. In the above example, suppose that  $t_1$  and  $t'_1$  were the closest time stamp matches to a given query time  $t$ . Then the pointers  $p_1$  and  $q_1$  would give the data stream positions associated with time  $t$ .

Alternatively, given a position in a particular data stream, the closest data pointer stored in the index can be identified. The time-stamp for that entry can then be compared to the time-stamps for the other data streams, resulting in a set of corresponding entries in the index. The data pointers for these entries can be followed, giving the corresponding positions in the other data streams. For example, suppose that  $p_0$  was the closest data pointer to a given query position. Then since  $t_0$  is matched to  $t'_0$  (after appropriate event filtering and data pointer interpolation), it follows that  $q_0$  is the position in the second data stream that corresponds to the query position. This could be used in the transcription example to synchronize the display of the transcript with the playback of the audio. It could also be used for multimedia indexing. In this case text queries would produce matches in the transcript text which could then be linked directly to segments of audio content.

Furthermore the recorded content can be modified in a post-processing phase to incorporate the cross-links produced by time-stamp matching. In the transcription system, for example, words in the transcript could be automatically hyperlinked to spoken words in the recorded data, using the capability for indexing into a data stream provided by formats like RealAudio. Or frames in a video sequence could be automatically labeled with text from a correlated transcript.

It should be clear that the invention could be adapted to many different scenarios. For example, by changing the time interval used in pointer interpolation we can address both closed captioning applications,

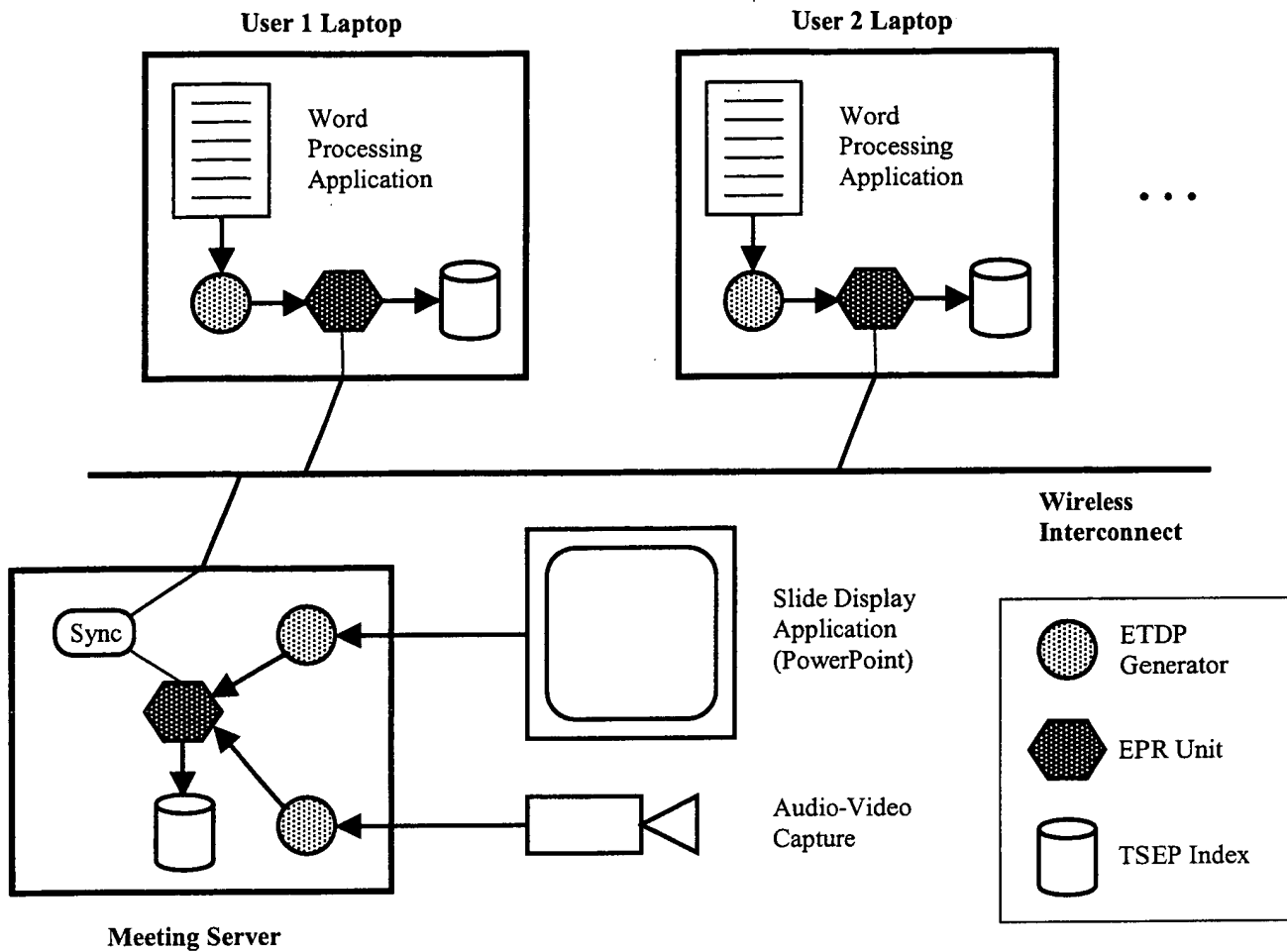


Figure 2: System architecture for distributed presentation recording system, based on the core invention of Figure 1.

where many words map to a single video frame, and transcription applications where words map to audio samples at a much finer level of granularity. Figure 2 shows the architecture for a presentation-recording system, which is a second application that is enabled by our invention. In this system, a multiplicity of users watching a presentation take notes using an application such as a word processor. In our preferred embodiment, each note-taker is using a portable computer, such as laptop, with a wireless network connection. Each portable is running a subset of the architecture in Figure 1, consisting of a data pointer generator for the word processor and a recording unit. In this case, each recording unit is only logging one stream of event-pointers. All of the recording units are synchronized via the wireless network so their time-stamps will be consistent. Here text events might be generated at the level of paragraphs or sentences rather than words as in the transcription example. Note that the issue of handling insertions and deletions remains.

The presentation system includes a second instantiation of our architecture from Figure 1, which runs on a meeting server. It has two generator modules, which service a slide presentation application, such as MS PowerPoint, and an audio-visual capture device. The event triggers for the PowerPoint application would be either a slide change or a within-slide animation. The data pointers would be the slide number or an identifier for the within-slide animation. The presentation module makes it possible to correlate notes against the particular slide or animation displayed by the presenter. Note that out-of-order access to slides is also present in this application, analogous to out-of-order generation of text entries in the note-taking

module.

Events for the audio-video capture device could be very simple, possibly just indicating the beginning and end of the presentation. Implicitly generated events and data pointers would provide pointers into the recorded stream at regular, pre-defined intervals. The recording unit for the meeting server is synchronized with the individual recording units on each user's portable computer, to ensure consistent time-stamps.

After the talk is over, each user would have the option of correlating their notes against both the audio-visual recording and the time-stamped pointers into the presentation slides. This could be done via a simple web interface. It could be performed by doing a merge sort between the user's event-pointer index and the one produced on the meeting server. The same web interface could then allow each user to retrieve a presentation slide and video clip associated with each sentence or paragraph in their notes. In addition, the user could replay the presentation recording, with both the PowerPoint slides and their recorded notes scrolling in synchrony with the audio-visual playback.

In a further embodiment, the notes from all of the users in attendance could be pooled into one larger time-matched index, by simply merge sorting all of the separate index files. The result could be easily cross-linked against the presentation slides and audio-video recording as described above. A searchable text index of the pooled notes could then be constructed, using AltaVista search technology, for example. This would allow people who had not attended the talk to search for keywords in the pooled notes. Each hit would have a set of cross-linked data pointers that would make it possible to retrieve the slides and video clips of the presentation associated with the keywords. The retrieval quality of this index could be further improved by augmenting the index with text from the slides themselves, as well as transcripts of the presentation obtained through speech recognition.

A key feature of this invention is that because we are explicitly storing data pointers we can cross-link data streams which do not share a common clock, or even have a consistent real-time clock. For example, if one stream consists of video or audio data it is not necessary that the other streams be expressible in terms of video frame rate or audio sample rate. Furthermore, our invention supports data streams in which the data rate can change dramatically in an asynchronous or unpredictable fashion.

#### **4.3.1 Event-Triggered Data Pointer Generators**

The Event-Triggered Data Pointer Generators detect trigger events from, for example, a user input application or a streaming data recording or playback device. The trigger events are prespecified and can be explicit or implicit. Explicit trigger events include, for example, an auto-save event in a word processing application, or a 'stop', 'play', 'record', or 'pause' command on a streaming data recording/playback device.

User input data is often created using a PC application such as a word processor. To get more resolution for the user input data, the Event-Triggered Data Pointer Generator can incorporate a gateway application to intercept keyboard and mouse inputs before they are passed to the user input application. Most operating systems enable a keyboard intercept functionality. For example, the Windows95 operating system has the **VKD\_Filter\_Keyboard\_Input** keyboard input service to allow a program to access all keyboard input before it is processed. This approach would allow trigger events to resolve down to the individual keystrokes.

Because of the nature of the streaming data recording/playback devices, only the 'play', 'record', 'fast forward', 'rewind', 'pause', and 'stop' events need be explicitly communicated to the Event-Triggered Data Pointer Generators. Given a fixed playback or record data rate, one embodiment of the Event-Triggered Data Pointer Generator could automatically generate data trigger events based on elapsed time, thus, implicit trigger events occur for every predetermined, fixed, interval of time. In this embodiment, the Event-Triggered Data Pointer Generators also must implicitly generate the data pointers, resulting in a

possible reduction of accuracy. To improve the resolution of links to the streaming data events, the Event-Triggered Data Pointer Generators for such data typically will trigger at a much higher rate than the Event-Triggered Data Pointer Generators monitoring the user input application.

For each trigger event, the Event-Triggered Data Pointer Generator produces a data pair consisting of an event label coupled with a pointer to the data that generated the trigger event.

#### **4.3.2 Event Pointer Recording Unit**

The Event Pointer Recording Unit gets data pairs from the Event-Triggered Data Pointer Generators, time-stamps each pair, filters the events and interpolates the data pointers as needed, then sorts the events according to chronological order, and records the events that are relevant to the linking between data streams. Events that can be deduced from other recorded events do not have to be recorded.

Linking between user input data events and the streaming data events is implicit and based on proximity in time. This linking has a fairly coarse resolution, within a few seconds, depending on the reaction speed of the user and/or the user input application. For the case of speech recording or playback, the time alignment can be further refined through the use of speech recognition [Pat5]. Another way to refine the speech alignment is to back up the user input event index to the beginning of a sentence by detecting pauses in the recorded audio.

#### **4.3.3 Implementation**

One possible embodiment of the core of our invention (the architecture of Figure 1) is as a software library with associated API that would allow any PC or handheld computer to function as an annotation device. Using the API, the user could interface to any standard application such as Microsoft Word as well as to video capture cards. The API would also define an interface to search technology such as AltaVista Personal Edition that would make it easy to search a set of recorded annotations. Our invention can be viewed as extending the utility of standard retrieval technology by providing a means for a user to link searchable attributes to a media stream that would otherwise remain inaccessible to indexing techniques. Using this invention, many specialized systems for tasks such as court recording could be replaced by PC systems running a general purpose software library with an off-the-shelf capture card.

For example, one data stream might consist of a series of words typed into a document editor such as Microsoft Word97. These words would comprise a set of notes for a presentation, captured in real-time during the talk. A second data stream might be an audio and video recording of the presentation. Using our invention, it is possible to easily establish the correlation between words in the notes and audio/video samples within the data multimedia stream.

For another example, this system can be used to automatically link user-entered meeting notes with the relevant sections in an audio recording of that meeting. The system could also be used to get a rough alignment between the transcript and the audio or video recording [Pat5]. In education and surveillance applications, a user could attach their own searchable annotations to course materials and captured footage. Many of these applications can gain considerable leverage from the ability to rapidly search the user input which has been associated with a media stream using standard indexing and retrieval technology. Our invention can be viewed as extending the utility of standard retrieval technology by providing a means for a user to link searchable attributes to a media stream that would otherwise remain inaccessible to indexing techniques.

The method described is one of the components of a semi-automatic system for producing off-line closed captions ([Pat5]).

#### 4.4. THE WHYS

Key differences between our invention and prior art are as follows:

- Our system links *time-correlated data pointers* generated by trigger events, instead of merely a description or identification of the trigger events. This approach accommodates the asynchronous nature of some media sources. It allows an arbitrary playback speed with a possibly non-streaming, non-continuous presentation clock. For example, in the closed captioning system disclosed in [Pat5], the audio playback rate could be sped up or slowed down to match the transcriber's typing rate. The audio playback device can also be paused, rewound, or fast forwarded during the transcription process. Because the Event-Triggered Data Pointer Generator provides a direct pointer (e.g. the tape counter or a recorded time stamp as opposed to an event time stamp) to the data being played during a trigger event, the transcription text is linked to the appropriate audio. Another part of the system then fine tunes the audio link to match the typed words with the spoken words. Because data linking is based on a recorded clock instead of a presentation clock, the audio clock does not necessarily have to be the master clock of the system, unlike most other systems.
- This approach also enables the synchronization of user input to media playback/recording devices that aren't tightly integrated into a unified system. It is well suited to systems that have little or no control of an external, asynchronous media stream. The Event Pointer Recording Unit only needs trigger event notifications from an external source in a timely fashion. The notifications must include a pointer to the appropriate point in the data when the trigger event occurred. For example, an external videotape player would have to provide its tape counter whenever a trigger event (any VCR command) occurred.
- This approach makes order of creation of the data streams unimportant. For example, when used in a text transcription system, our invention allows a user to create a transcript before, during, or after an audio recording. If a transcript is created before an audio recording, there must be a mechanism (e.g. user input) to generate trigger events to the text during the subsequent recording.
- Once the trigger events are defined, they are *automatically* detected and *linked* without user intervention.

Our method is useful and efficient for the following reasons:

1. The system is mostly independent of the user input application. No modification to the application is required to provide for the generation of observable trigger events.
2. The trigger-event detection scheme is very general and adaptable. For example, if a word processing program is used for user input, the auto-save or auto-recovery mode could be set for the most frequent updates. The trigger event can then be the auto-save action. Higher resolution is possible with the use of a keyboard/mouse gateway application.
3. The linking operation is automatic. Once the trigger events are defined, operation of the system is transparent to the user.

This invention differs from the patented systems described above in the following ways:

- In *Arrangement for recording and indexing information* [Pat1], the linking of indexing information with a separate recording is based on the linking of the position of the hand-written indexing notes and the position of the recording in the recording medium. The invention described here links arbitrary user-input with arbitrary, concurrent data stream(s) via pre-defined, user-selected, time-stamped, events.
- *Interactive system for producing, storing and retrieving information correlated with a recording of an event* [Pat2], requires the user to explicitly request a 'time-zone' to link subsequent user input to a recording. The invention described here allows the user to predefine a trigger event to automatically link user input to the recording.

- *Method and apparatus for recording and analyzing an interaction log* [Pat3], describes a system to record various state parameters relating to a trigger events within an interactive system. The invention described here goes beyond this system by correlating pre-defined trigger events within a user input stream with one or more concurrent recordings.
- *Computer graphics data recording and playback system with a VCR-based graphic user interface* [Pat4], is a system limited to the recording and linking data from an X-windows session. The invention described here is independent of the X-windows environment and addresses the linking of two or more independent, concurrent data streams.
- *Computerized court reporting system* [Pat6], is a system limited to stenographic annotation of trial proceedings. The invention described here is independent of the courtroom annotation environment.

Our invention also differs from other prior art in significant ways:

- Microsoft Powerpoint97 offers a 'record narration' feature which enables the recording and playback of audio associated with a given presentation. Powerpoint97 can either imbed the audio within a presentation or save the audio explicitly as a external WAV format file that is linked to the presentation. Each slide within the presentation can only have one embedded or linked audio file and this file overwrites all other audio associated with that slide (e.g. presentation sounds). Audio for a given slide can be inserted or deleted, but Powerpoint97 doesn't provide any tools to edit the audio at a finer granularity. The application also requires explicit user intervention to pause or stop the audio recording during a presentation. While our invention could be used to implement the Powerpoint97 'record narration' feature it is much more flexible. Unlike the Powerpoint97, our system does not have to link the audio to a slide. Our system's cross-linking via a time-stamped event index avoids a master-slave relationship between the slides and the audio. With the Powerpoint97 feature, the slide must be created before recording the audio. Our system allows the audio to be recorded before the creation of the slide(s). Our system can also offer the following additional authoring options not possible with the current Powerpoint97 system:
  - The user can write slides and record audio simultaneously.
  - By editing the event index, It is possible to :
    - Add a new slide to a previously recorded audio-slide presentation without necessarily disrupting the audio with a slide that is silent.
    - Without explicitly editing the audio file, the user can insert more audio into a previously recorded audio track. The new audio can either displace or overwrite the previous audio. In addition, if the presentation output device supports multiple audio channels, new audio can be output simultaneous to the previous audio.
- The Unix utility, syslogd, enables the time-stamped logging of user-defined events. With little or no modification, this utility could serve as the Event Pointer Recording Unit of the invention as described above, however syslogd alone does not cover our invention. Cross-linking and indexing trigger events are the crucial parts our invention missing from syslogd. The event trigger detection units provide a means to create a direct pointer to the data producing the trigger event. For example, if our invention was used in a system to transcribe an audio recording, the event index output of the Event Pointer Recording Unit could be used to produce a hyperlinked document with each word in the transcription linked to the place in the audio recording that the transcriber heard when he was typing the word. With speech re-alignment process described in [Pat5], the hyperlinked document could be refined so that the typed word would link directly to the spoken word.

In general, this prior art has failed to capture the essential architecture of an annotation which is embodied in our invention. As a result these prior inventions were limited in both the contexts in which they could be employed and the type of annotations that could be created. Furthermore, our invention anticipates the

widespread availability of efficient text-based indexing and retrieval technology such as embodied in AltaVista Personal Edition. We designed our invention so as to be easily integrated with this search technology.

## **5. Related Inventions**

### **5.1. PATENTS**

[Pat1] *Arrangement for recording and indexing information*, by Diane J. Rindfuss, US patent 04841387. Jun, 20, 1989.

[Pat2] *Interactive system for producing, storing and retrieving information correlated with a recording of an event*, by Karon A. Weber, Alex D. Poon, and Thomas P. Moran, US patent 05564005. Oct. 8, 1996.

[Pat3] *Method and apparatus for recording and analyzing an interaction log*, by Toshiyuki Asahi and Hidehiko Okada, US patent 5793948. Aug. 11, 1998.

[Pat4] *Computer graphics data recording and playback system with a VCR-based graphic user interface*, by John Trueblood, US patent 5748499. May. 5, 1998.

[Pat5] *An efficient method for producing off-line closed captions*, by JM Van Thong and Michael Swain. US patent, to be filed.

[Pat6] *Computerized court reporting system*, by John C Jeppesen, US patent 4924387. May. 8, 1990

[Pat7] *A speech rate control method using speech recognition*, by JM Van Thong and Davis Pan. US patent, to be filed.

### **5.2. PRODUCTS**

No current products use the technology described by this invention.

### **5.3. JOINT RESEARCH**

This work was solely done at the Cambridge Research Laboratory.

## **6. Commercialization/Publication**

This concept has not been discussed outside of Compaq nor published in any way. There are no immediate plans to productize this invention.

## **7. Who and Where at Compaq**

1. Davis Pan, pan@crl.dec.com, (617) 551-7652

2. Jim Rehg, rehg@crl.dec.com, (617) 551-7645

### **7.1. YOUR SUPERVISORS**

Manager: Rishiyur Nikhil, CRL Acting Lab Director, (617) 551-7621

### **7.2. YOUR GROUPS AND BUS**

Davis Pan and Jim Rehg are members of the Compaq Cambridge Research Laboratory (CRL), CRG, and ATG.

**8. Product Use**

There are currently no Compaq products embodying this invention. This invention is part of research's effort to develop new fundamental technologies for Compaq. A current product, Mediavista, could incorporate this technology in the future. The invention can be used to aid the alignment of an audio or video recording with closed captioning. The Mediavista search engine could then allow more accurate media searches based on the closed captioned text. A possible future product, a digital video recorder, could also use this technology. The invention could also be incorporated in a stand-alone product, for linking real-time user data (for example annotations or transcripts) to a real-time data recording (for example an audio or video recording).

**9. Signatures**

Davis Pan

Jim Rehg